A2W-Macro

Long strings can overflow stack

Sean Barnum, Cigital, Inc. [vita[1]]

2005-10-03

Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 6430 bytes

Attack Categories	• Malicious Input • Denial of Service
Vulnerability Categories	• Multibyte Character • Unconditional • Unhandled Exception
Software Context	• String Conversion MACROS
Location	
Description	A2W macros can lead to stack overflows or unhandled exceptions. The Microsoft Active Template Library (ATL) is a set of template-based C++ classes that simplify the programming of Component Object Model (COM) objects. It provides the A2W set of macros for converting between ASCII and wide (Unicode) characters. The A2W macros call _alloca(), which allocates memory from the stack. If the input string is too long, so that the stack would overflow, _alloca() will throw an exception. If the exception is not caught, the program will halt.
APIs	see table below

FunctionName	Comments
A2W	
CW2CT	
W2A	

Method of Attack	An attacker can provide very long strings of input to vulnerable methods potentially gobbling up all the stack space. If the appropriate exceptions are not caught, the program will halt causing a DoS.
Exception Criteria	
Solutions	

Solution Applicability	Solution Description	Solution Efficacy

1. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/35-BSI.html (Barnum, Sean)

	Whenever A2W macros are used.	Avoid use of these methods. New conversion macros were introduced in ATL 7.0. These macros, which have somewhat different usage, are more robust and do not allocate memory on the stack. Use these instead of the older A2W macros. If you can't use the ATL 7.0 conversion macros, always wrap the conversion with an exception handler. Reset the stack if stack overflow exceptions occur. MSDN advises "Check the length of the strings before passing them to these macros to avoid potential buffer overrun problems. Stack overflows are exceptions that could also be caught with try/ except." The length of string that would be problematic depends on how the / STACKSIZE linker option is used. It is not clear how	Believed to be effective.

		stack overflows would occur for these macros. Possibly this recommendation refers only to the older macros. Check the input length before using any of these macros to ensure that lengths are not dangerously large	
Signature Details	Presence of any of the A2W macros identified above.		

Examples of Incorrect Code	`// Note use of older macro that allocates stack memory.`
	`// Does conversion, but could overflow stack and throw exception`
	
	`LPCTSTR szr =`
	`A2T( szReplaceFile );`		

Examples of Corrected Code	`// Use new macro`
	`// Note form of code above doesn't work for new macros`
	
	`if (strlen(szReplaceFile) > MAX_REASONABLE_SIZE) { /* handle error */ }`
	`else {CA2TEX`
	`szr( szReplaceFile );}`
	
	`// If must use older macro, catch exception and reset stack`
	
	`if (strlen(szReplaceFile) > MAX_REASONABLE_SIZE) { /* handle error */ }`
	`__try {`
	`LPCTSTR szr =`
	`A2T( szReplaceFile );`
	`// use szr`
	`} __except`
	`((EXCEPTION_STACK_OVERFLOW == GetExceptionCode()) ?`
	`EXCEPTION_EXECUTE_HANDLER :`
	`EXCEPTION_CONTINUE_SEARCH) {`
	`_resetstkoflw();`
	`}`		

Source Reference	Howard, Michael. Tackling Two Obscure Security Issues[2] (2002).
Recommended Resources	• MSDN reference for ATL and MFC String Conversion Macros[3] • MSDN TN059: Using MFC MBCS/Unicode Conversion Macros[4]
Discriminant Set	**Operating System**: • Windows **Languages**: • C • C++

Cigital, Inc. Copyright

1. mailto:copyright@cigital.com